Dynamic Sublists: Scaling Unmoderated Mailing Lists

Ellen Spertus – Mills College
Robin Jeffries – Sun Microsystems, Inc.
Kiem Sie – Mills College

ABSTRACT

Unmoderated electronic mailing lists suffer from the paradox that the more successful they are, the more difficult they are to use and administer because of the increasing number of users and discussions. Traditional solutions to this problem, such as creating static sublists or providing a Web interface, are not always desirable. We discuss the problems with traditional solutions and present dynamic sublists, an approach to scaling online communities that increases communication opportunities without overwhelming users or administrators. We have built a system, Javamlm, implementing dynamic sublists, and report on its implementation, use, and future.

Motivation

Systers [2] is an unmoderated email list for women in computer science, originally created in 1987 for 12 women. The list has been tremendously successful at creating a community for people who had previously felt isolated and has grown to include over 2300 members in 38 countries. As Systers has grown, however, it has lost hundreds of members, especially senior women, because of the increased message volume that has come with increased membership. Additionally, members have felt less free to post to the entire list, instead responding directly to the sender of a message or not at all, reducing the sense of community and the utility of the list. Our goal is to provide a better format for such a large community than an unmoderated email list, while keeping the features that make Systers successful.

Rejected Alternatives

A common general solution to heavy volume on mailing lists is moderation, which can be performed top-down by a small set of administrators or bottom-up through collaborative filtering. While the most obvious cost of moderation is the human overhead required, a more fundamental problem is that moderation schemes limit diversity through the "tyranny of the majority." A topic of little interest to the majority may be of great interest and value to a minority. A better scheme (which we describe below) would allow users to customize which messages they see.

A conventional alternative to moderation is the creation of static sublists interested in particular subtopics. For Systers, there could be technical groups focused on Web design, groups discussing how to best deal with maternity leave, or support groups for pretenure academics. However, while an individual Syster may not be interested enough in maternity leave to join a group that discusses that exclusively, part of what members find valuable is hearing about other women's issues in a wide range of areas (especially ones that might affect her in the future). Thus, completely eliminating the sharing of information about maternity leave with the broader group lessens the value of the list. In the limit, there would be no common discussion among the entire group, just a vast collection of special interests. For these reasons, we rejected the idea of relying primarily on static sublists.

The most obvious way to support customized views is to go to a Web-based, newsgroup-like format, where Systers can browse the topics that interest them. Because it is not email-based, this solution would be unacceptable to many long-time Systers, who have expressed a preference for email, as well as to members in poorer countries with less Internet connectivity. It also removes the immediacy of the information, since most Systers would check the website at most daily or weekly, which would change the feel of the community.

Our Approach

We have a solution that we believe meets many of our design goals: dynamic sublists, which are similar to Usenet threads. Members are able to easily create, subscribe to, or unsubscribe from dynamic sublists. The opening message of a dynamic sublist is sent to all members. When a member signs up for Systers, she specifies whether she wants to see all messages or only the first message of each thread (a less precise but more user-friendly synonym for "dynamic sublist"). The default behavior is for users to be subscribed to all threads in order to mimic the behavior of the current system. Users may also specify whether they prefer to receive messages as plain text or as html.

Threads could exist for a limited period of time or be permanent. Under the current system, a volunteer collects job listings and posts them to the list once per week. She also forwards them immediately to

Systers who are looking for employment. Having a *systers-jobs* thread would eliminate the human administration costs while increasing functionality. An individual could specify any of the following behaviors:

- Receiving each job posting immediately
- Receiving digests of job listings on a daily, weekly, or monthly basis
- Never seeing job listings

At any time, she could change her preference without going through a human administrator.

Implementation

We have built a prototype, the Java Mailing List Manager (Javamlm), which implements dynamic sublists. Figure 1 shows the structure of the system. Information about users, threads, and messages is kept in a relational database. We are currently using Postgres as our database management system.

Our mail transport agent (MTA) is qmail, which has support for user-controlled mailing lists. Specifically, when incoming mail contains a hyphen in the local portion of the address (e.g., systers-subscribe@javamlm), the string before the hyphen is interpreted as the username, and the string after the hyphen is handled as specified by the user. In the case of Javamlm, qmail the message is piped to a Java program, which handles it as appropriate.

We have also built a Web interface to the database for users and administrators using AOLserver and Tcl. All of these tools are open source, which will allow us to release the entire package as open source.

Schema

The database schema is shown in Figure 2. The Subscriber relation contains information about each individual subscriber, such as name, email address, preferred message format (plain text or html), and whether or not to be subscribed to new threads. The Thread relation contains information about each thread, including a unique name and a reference to the initial message, which is represented in the Message relation, which stores meta-data about each message.

Because the Thread and Message relations reference the Subscriber relation, we cannot delete a record from Subscriber if a user unsubscribes. Instead, Subscriber has a boolean field, deleted, which is set to true if a user unsubscribes, inhibiting message delivery.

When a new thread is begun, the initial message is sent to all current subscribers. The SQL query is:

```
SELECT text_format, email
FROM Subscriber
WHERE deleted = FALSE
```

Subsequent messages within a thread are only sent to members who have expressed a desire to see them. A user's default preference for new threads is represented in Subscriber.preference. A value of 0 means "not subscribed to new threads," while 1 means "subscribed to new threads."[1] To honor a user's default preference, the SQL query for subsequent messages within a thread would be:

```
SELECT text_format, email, preference
FROM Subscriber
WHERE deleted = FALSE
AND preference = 1
```

The Override relation is used when a subscriber wants to override his or her ordinary preference for a specific thread. For example, if subscriber 99 ordinarily does not receive subsequent messages in threads (Subscriber.preference = 0), the following entry in the Override relation would indicate that she wishes to receive all messages in thread 157:

subscriber_id	thread_id	preference
99	157	1

Similarly, an Override.preference value of 0 indicates

[1]We chose to use an integer, rather than a boolean, representation, for expandability.

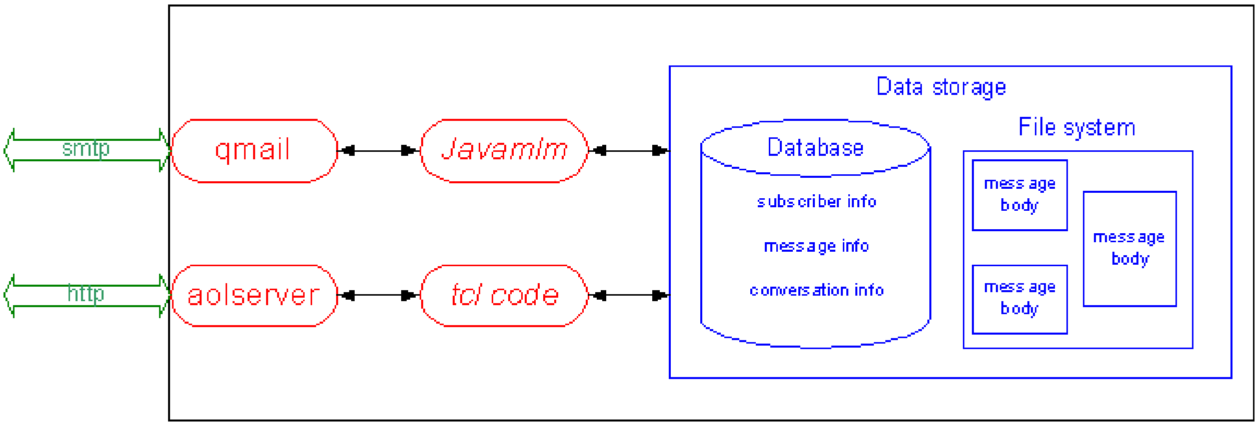

Figure 1: The structure of the system. Protocols are shown in arrows, processes in ovals, and store data in the box on the right. AOLserver and qmail are preexisting software packages. The authors' contributions are Javamlm, which provides mail-based access to the system and the tcl code, which is used for http-based access.

that a subscriber does not want to see further messages in a thread.

This is the complete SQL query to determine to whom a subsequent message in thread 37, for example, should be sent:

```
SELECT text_format, email
FROM Subscriber
WHERE deleted = FALSE AND
((Subscriber.preference = 1 AND
  NOT EXISTS
    (SELECT * FROM Override
     WHERE Override.subscriber_id =
     Subscriber.subscriber_id
     AND Override.thread_id = 37
     AND Override.preference = 0))
 OR
 (Subscriber.preference = 0 AND
  EXISTS
    (SELECT * FROM Override
     WHERE Override.subscriber_id =
     Subscriber.subscriber_id
     AND Override.thread_id = 37
     AND Override.preference = 1))
)
```

Messages

The bodies of messages are stored as regular files rather than in the database for the following reasons:

- Messages' highly variable size makes database storage awkward or inefficient.
- Locked access to messages is not needed, because messages are written once and never modified.

Subscriber

field	type	notes	sample value
subscriber_id	INTEGER	primary key	10328
email	VARCHAR(255)		"borg@iwt.org"
firstname	VARCHAR(16)		"Anita"
lastname	VARCHAR(16)		"Borg"
preference	INT2	Subscribed to new threads?	0 (no) 1 (yes)
format	INT2	Format to receive messages	1 (ASCII) 2 (HTML) 3 (both)
password	CHAR(16)	encrypted	FS9_eA%6
deleted	BOOLEAN	Has user unsubscribed?	false

Thread

field	type	notes	sample value
thread_id	INTEGER	primary key	157
thread_name	CHAR(16)	unique	fellowships51
sender_id	INTEGER	Person.subscriber_id	10328
base_message_id	INTEGER	Message.message_id	12000
subject	VARCHAR(255)	Subject of initial message	"AAUW fellowships"
status	INT2	Status of thread	0 (new) 2 (closed) 1 (in progress) 3 (perpetual)
parent	INTEGER	Thread.thread_id, can be null	null

Message

field	type	notes	sample value
message_id	INTEGER	primary key	12000
sender_id	INTEGER	Person.subscriber_id	10328
thread_id	INTEGER	Thread.thread_id	157

Override

field	type	notes	sample value
subscriber_id	INTEGER	Person.subscriber_id	10328
thread_id	INTEGER	Thread.thread_id	aauw23
preference	INT2	Only used to override default subscriber preference 0 (subscribed) 1 (unsubscribed)	

Figure 2: Database schema. The Subscriber relation contains information about each individual subscriber, such as the email address and whether or not to be subscribed to new threads. The Thread relation contains information about a given thread. Meta-data about a message is stored in the Message relation; the actual body is stored separately in a file. The Override relation is used when a subscriber wants to override his or her ordinary preferences for a specific thread.

- The file system provides an easier interface than the database for separate programs to access archived messages.

Javamlm accepts messages formatted as plain text, html, or both (through MIME multipart). Every message is converted, if necessary, to each of these formats, so users can receive messages in the format they prefer. Html is converted to text by the Lynx browser, which provides a command line option "-dump" for this purpose. Conversion from plain text to html is done through a modified version of the Perl script otxt2html, written by Ola Lundqvist [11]. Once converted, the messages are stored to disk. For example, the body of message 120 to Systers would be stored in the files:

- ˜systers/javamlm/archive/120.text-plain
- ˜systers/javamlm/archive/120.text-html

MIME-compliant messages are assembled through calls to the javax.mail package. To accommodate all of the users' preferences, up to six different messages are created: in each of the three formats (text/plain, text/html, and multipart) and with two different footers (unsubscription and subscription information), which are described more in the next section.

The program iterates through each of the subscribers, choosing the correct message format, specifying the email address, and creating a variable envelope return path (VERP). VERPs, invented by Dan Bernstein, automate matching bounced emails with subscriber addresses [2]. For example, a message to ada@lovelace.com on the Systers list would have a VERP of systers-error-ada=lovelace.com@javamlm.mills.edu. This is placed in the SMTP "From" field [9], which instructs mail transport agents (MTAs) where to send error messages. No matter how many times the message is rerouted through ".forward" files and other mechanisms [15], if it eventually bounces, the subscriber information will be intact, allowing the subscriber to be retried or unsubscribed. Note that message headers [12] are not personalized for each subscriber, only SMTP headers [9], minimizing the per-subscriber overhead.

User Interface

Figure 3 shows the Web form for joining a list. Other forms allow users to change their membership options and allow the administrator to access and modify members' settings. Web access is password-protected, using cookies. Currently, messages cannot be sent or viewed through the Web.

Users' primary method of access is through email. A member of Systers creates a new thread by sending the introductory message to systers-new@javamlm.mills.edu. Alternately, if she wishes to name the thread "fellowships," for example, rather than having a system-assigned name, she would send her introductory message to systers-new-fellowships@javamlm.mills.edu. All members would then receive the first message in this new thread. Figure 4 shows a sample message as it appears to the sender and an html-enabled recipient. Note that a number has been appended to the thread name, "fellowships," to make it unique.

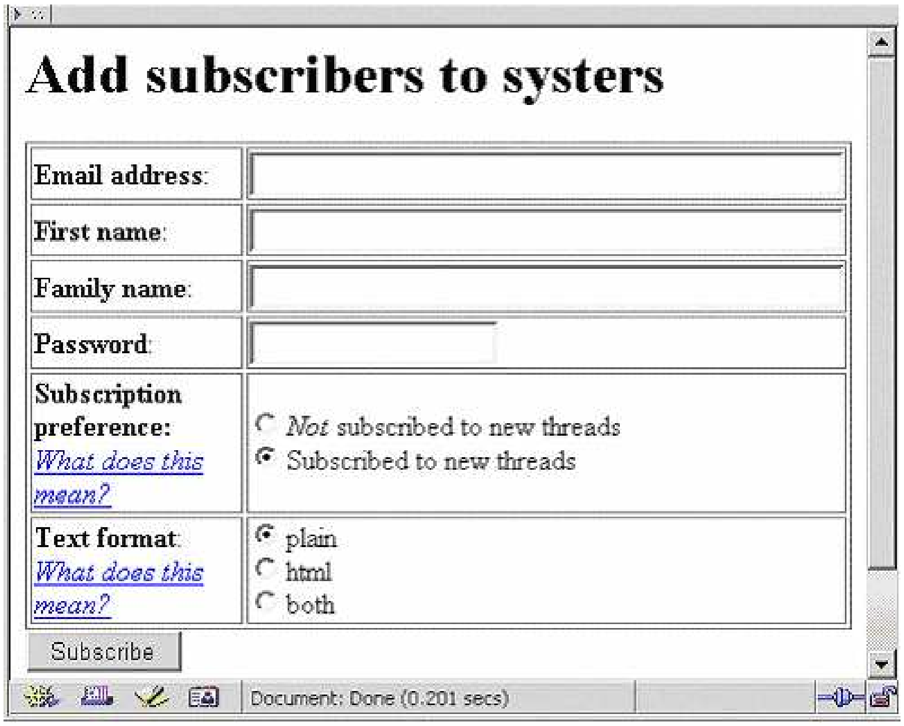

Figure 3: Sign-up screen.

If a recipient wishes to reply to the sender, she uses her MUA's[2] "reply" function. If she wishes to contribute to the thread, she uses the "reply-all" function.

Unless a recipient specifies otherwise, her default preference (e.g., "view all messages in new threads") would be in effect for subsequent messages in the new thread. There are two ways of overriding her default preference, by email or through the Web. As Figure 4 shows, instructions for unsubscribing from the thread appear at the bottom of the message to users who are by default subscribed. Similarly, if a user is by default not subscribed to new threads, instructions for subscribing are included.

Unsubscribing by email takes at least two steps: activating the embedded mailto link (e.g., mailto:systers-unsubscribe-fellowships51@javamlm.mills.edu) and confirming to the MUA that the email should be sent. (As RFC 2368 points out, it would be insecure for a MUA not to require user confirmation [7].) To minimize the number of required user actions, we also provide in each message a Web interface for overriding one's default preference, as shown in Figure 4 (e.g., http://javamlm.mills.edu/scripts/override?listname=systers&thread=157&preference=0). Note that the *subscriber_id* is not included in the URL, for three reasons:

1. It would be inefficient. Including a different *subscriber_id* in each message would require a unique Java MimeMessage object for each recipient.
2. It would be insecure. If the *subscriber_id* were embedded, it would be easy for a subscriber to unsubscribe other people by substituting their *subscriber_id*s.
3. It is not necessary. Because the Web interface uses cookies to store the *subscriber_id* of users who have successfully logged in, it is not necessary to reenter it. If there is not a valid cookie, the user is prompted for her username and password. Users who share computers can explicitly log out of the system.

Experience

We performed a month-long user test with 20-40 members of the main Systers list. Unfortunately, we were unable to generate a critical mass for sustained thread. Still, we were able to extract some useful information.

Of the 40 subscribers, 39 chose to receive all messages by default (i.e., preference = 1). Twenty-six members specified plain text as their preferred format, six hypertext, and eight both (i.e., MIME multipart). Nobody unsubscribed from the list during the trial period. We were surprised that the majority of users chose plain text over hypertext, underscoring the importance of having a good ASCII user interface.

Seven of the 11 threads were created by the system's authors. In an attempt to encourage members to use the unsubscribe feature, we created a thread, called *word1*, that forwarded the daily word from Anu Garg's A-Word-A-Day service (http://wordsmith.org/awad). Six members unsubscribed from the thread without difficulty. One member publicly responded positively to the thread with her thoughts on a word,

[2]A mail user agent (MUA) is a client program for a user to access and compose email. Common MUAs include Eudora, Outlook, RMAIL, and pine.

```
To: systers-new-fellowships@javamlm.mills.edu
Subject: AAUW Fellowships
From: spertus@mills.edu

Information about AAUW fellowships is now
available at http://www.aauw.org.
```

Figure 4(a): Initial message.

```
To: systers-fellowships51@javamlm.mills.edu
Subject: SYSTERS: AAUW Fellowships
From: borg@javamlm.mills.edu
Senders: systers-fellowships51@javamlm.mills.edu

Information about AAUW fellowships is now available at
http://www.aauw.org.

To unsubscribe from this thread, send email to
systers-fellowships51-unsubscribe@ javamlm.mills.edu or visit
http://javamlm.mills.edu/scripts/override?listname=systers&thread=157&preference=0.

To unsubscribe from systers, send email to
systers-unsubscribe@javamlm.mills.edu.
```

Figure 4(b): Message as viewed by recipient.

Figure 4: The first message in a new thread, as (a) written by the sender and (b) seen by the recipient. This assumes that the recipient has hypertext enabled and is by default subscribed to new threads. The recipient will continue to receive further messages in this thread (i.e., sent to systers-fellowships51@systers.org) unless she activates the mailto orhttp link to unsubscribe from the thread or from thelist.

which motivated one of the authors to create a new thread asking members for theirs pet peeve about word misuse. It was unclear how to begin this new topic. If it were within thread *word1*, subscribers could not avoid the discussion without unsubscribing from the A-Word-A-Day message, which many enjoyed. Instead, we created a brand-new thread, *peeves1*, which had the disadvantage of going to people who had unsubscribed from *word1*. This showed the desirability of allowing for threads to have children, which would allow child threads to begin by going only to subscribers of an existing thread instead of to all users.

The first message in the *peeves1* thread was created by sending mail to systers-new-peeves@javamlm, which was cc'd to the subscriber who had first expressed her opinion. Because this member received the message directly (and not just through Javamlm), the to-line contained systers-new-peeves@javamlm instead of systers-peeves1@javamlm. Thus, when she replied, another thread, peeves2, was created. We are exploring solutions to this problem.

A point that no users raised but that concerned the authors was how to deal with individuals' joining the list while a thread is in progress. If a user is by default unsubscribed to threads, she will not see any messages until a new thread begins (which may be the right behavior). We may want to provide new users with the options of being filled in on active threads, perhaps through a Web interface.

A related problem is what to do if a user unsubscribes from a thread (either by default or explicitly) and then explicitly subscribes. Currently, she would never get the messages that occurred while she was not subscribed. This is a particular problem for users who are unsubscribed from new threads by default, since, by the time they explicitly subscribe to a thread of interest, earlier messages may be lost to them. Clearly, we need to design a better approach.

Another problem that arose was varying hypertext production by different MUAs. The Eudora mailer encloses a formatted message with ''html'' tags, while Yahoo! Mail does not. Javamlm failed to properly insert (un)subscription information in messages formatted differently from expected.

One user requested that subscribers be emailed their user name and password, a feature provided by Mailman [14]. We had opted not to provide this feature both because of the insecurity of email and because we only store encrypted passwords. Clearly, some users prefer convenience to greater security, suggesting that we should provide per-user or per-list security options.

Related Work

While many people have addressed minimizing administration cost for owners of large mailing lists [e.g., 1, 4, 5, 14], our goal is more to minimize costs for *users* of high-volume mailing lists without increasing administrative overhead.

Our system is a direct descendant of the threaded news reader (trn) for Usenet [8] by Wayne Davison, which grouped messages with common ancestors into threads, allowing users to read articles by thread, instead of by date or subject, or to automatically avoid further messages in the thread. While the introduction of threads revolutionized newsreaders, it has been slow to cross over to email lists.

Ka-Ping Yee built a system, Roundup [16, 17, 18], supporting ''fine-grained mailing lists,'' making use of the ''In-Reply-To'' header [12]. He used the term ''issue'' for what we call a ''thread'' or ''dynamic sublist.'' Each issue has a corresponding set of users who receive new messages, called ''nosy lists.'' Users are added to an issue's nosy list if they are found in the ''From,'' ''To,'' or ''Cc'' headers of a message within the issue.

Jamie Zawinski has implemented and documented [19] an algorithm for threading email messages based on the ''In-Reply-To'' [12] and ''References'' [8] headers. Mark Crispin and Kenneth Murchison recently described the algorithm more formally and proposed changes to the IMAP protocol for server-side support for grouping messages into threads [6]. While making the best of the current situation, this approach is limited by MUAs' and users' inconsistent use of the ''In-Reply-To'' and ''References'' headers. For example, there's nothing to stop a user from manually copying the ''to'' field in replying to a message, rather than using a MUA's reply-to function. By embedding thread information in the ''to'' address, our system forces MUAs and users to include it in replies. Combined with Crispin and Murchison's proposed IMAP extensions, more powerful server-side threading could be provided.

Future Work

We consider our current system, Javamlm, to be a prototype, which we do not plan to extend. Instead, we will incorporate support for dynamic sublists into Mailman, the GNU mailing list manager [14]. We chose Mailman because it has the following features:

- Web-based interfaces for both administrators and users.
- Per-list and per-user configurability, including digesting.
- Automatic web-based archiving.

We plan to extend each of these features, such as allowing users to specify that a specific dynamic sublist should be delivered in digest form, while others should be delivered immediately. This would enable the *systers-jobs* example described earlier, allowing job-seekers to receive announcements immediately and other users to receive announcements as weekly digests or not at all.

Unlike the *jobs* sublist, most dynamic sublists are expected to be active for a short period of time (days or weeks). Planned future functionality would give users the additional option of seeing only a sublist's first message and a final summary message created by the thread's originator. We also plan to support individual keywords so a subscriber can automatically see all messages containing ''linux'' and ''administration'' if she so specifies.

We will also address the issues raised from our user test, such as the best means for creating child threads, supporting mid-thread subscription, and per-user security options. We welcome further input and participation in the project by anyone interested.

Acknowledgments

We are grateful to Sara Kiesler, Jennifer Goetz, and Lee Sproull for performing a study of Systers members with Robin Jeffries, upon which this work is based. We would also like to thank Anita Borg and the Systers members who participated in those surveys and informal discussions, offered many ideas for redesigning of Systers, and volunteered their skills and effort to improve the Systers community. We have also received useful input from Gloria Montano, who led the first user test. Our understanding of email protocols was greatly enhanced by members of the *List Managers Mailing List*. We received encouragement and valuable feedback, including pointers to related work of which we had been unaware, from the LISA referees, Sigmund Straumsnes, and Mailman author Barry Warsaw. Ellen Spertus and Kiem Sie are partially supported by a National Science Foundation Faculty Early Career Development grant. This work is being done in cooperation with the Institute for Women and Technology, which hosts Systers.

References

[1] Bernstein, D. J., ''Ezmlm,'' http://cr.yp.to/ezmlm.html .

[2] Bernstein, D. J., ''Variable Envelope Return Paths,'' http://cr.yp.to/proto/verp.txt, February, 1997.

[3] Borg, Anita, "Why Systers?" *Computing Research News*, http://www.systers.org/keeper/whysys.html, 1993.

[4] Chalup, Strata Rose, Christine Hogan, Greg Kulosa, Bryan McDonald, and Bryan Stansell, ''Drinking from the Fire(walls) Hose: Another Approach to Very Large Mailing Lists, LISA XII,'' 1998.

[5] Chapman, D. Brent, ''How I Manage 17 Mailing Lists Without Answering '-request' Mail,'' *LISA VI*, 1992.

[6] Crispin, Mark R., and Kenneth Murchison, *Internet Message Access Protocol – Thread Extension, Internet Draft*, IMAP Extensions Working Group, IETF, 2001.

[7] Hofman, P. L. Masinter, and J. Zawinski, *The mailto URL scheme (RFC 2368)*, Network Working Group, IETF, July, 1998.

[8] Horton, A. and R. Adams, *Standard for Interchange of USENET Messages (RFC 1036)*, Network Working Group, IETF, December, 1987.

[9] Klensin, J., ed., *Simple Mail Transfer Protocol (RFC 2821)*, IETF, April, 2001.

[10] Lindberg, Fred, ''Ezmlm/idx Manual,'' version 0.32, http://www.ezmlm.org/ezman-0.32/index.html, March, 1999.

[11] Lundqvist, Ola, ''otxt2html.pl,'' http://www.opal.dhs.org/programs/otxt2html/index.oml .

[12] Resnick, P., ed. *Internet Message Format (RFC 2822)*, Network Working Group, IETF, April, 2001.

[13] Rosenthal, Chip, '' 'Reply-To' Munging Considered Harmful,'' http://www.unicom.com/pw/reply-to-harmful.html, May, 1999.

[14] Viega, John, Barry Warsaw, and Ken Manheimer, ''Mailman: The GNU Mailing List Manager,'' *LISA XII*, 1998.

[15] Westine, A. and J. Postel, *Problems with the Maintenance of Large Mailing Lists (RFC 1211)*, IETF, March, 1991.

[16] Yee, Ka-Ping, ''Roundup: A Simple and Effective Issue Tracker in Python,'' short talk, Eighth International Python Conference, 2000.

[17] Yee, Ka-Ping, ''Roundup: An Issue Tracking System for Knowledge Workers,'' design proposal, Software Carpentry Design Competition first round, March, 2000.

[18] Yee, Ka-Ping, ''Roundup: An Issue Tracking System for Knowledge Workers, Implementation Guide,'' Software Carpentry Design Competition second round, June, 2000.

[19] Zawinski, Jamie, ''Message Threading,'' http://www.jwz.org/doc/threading.html, 2000.